

**NAME:** Ahtisham Ahmed

**Roll no:** 014

**TASK:** No. 1

#include <iostream>

using namespace std;

class student {

public:

student\* next;

student\* prev;

student\* end;

student\* head = NULL;

int roll;

int count = 0;

};

class list: public student {

public:

student\* create\_node() {

student\* newnode = new student;

return newnode;

}

student\* find\_last() {

student\* temp = head;

student\* last\_node = NULL;

if (head == NULL) {

cout << "LAST NODE NOT FOUND[ERROR: LINKED LIST IS EMPTY]" << endl;

}

else {

while (temp->next != NULL) {

temp = temp->next;

}

last\_node = temp;

}

return last\_node;

}

void input\_data(student\* newnode) {

cout << "\nENTER THE ROLL NO : "; cin >> newnode->roll; cout << endl;

}

void insert\_at\_head(student\* newnode) {

student\* temp = head;

if (head == NULL) {

head = newnode;

newnode->prev = NULL;

newnode->next = NULL;

}

else {

newnode->next = temp;

newnode->prev = NULL;

head->prev = newnode;

head = newnode;

}

count++;

}

void insert\_at\_end(student\* newnode) {

student\* temp = head;

student\* last\_node = NULL;

if (head == NULL) {

head = newnode;

newnode->next = NULL;

}

else {

last\_node = find\_last();

last\_node->next = newnode;

newnode->next = NULL;

newnode->prev = last\_node;

}

count++;

}

void display\_from\_head() {

student\* temp = head;

if (head == NULL) {

cout << "NOTHING IN THE LIST TO BE DISPLAYED" << endl;

}

else {

while (temp != NULL) {

cout << temp->roll << endl;

temp = temp->next;

}

}

}

void display\_from\_end() {

student\* temp = head;

student\* last\_node = NULL;

if (head == NULL) {

cout << "NOTHING IN THE LIST TO BE DISPLAYED" << endl;

}

else {

last\_node = find\_last();

while (last\_node != NULL) {

cout << last\_node->roll << endl;

last\_node = last\_node->prev;

}

}

}

void search\_by\_roll(int roll\_search) {

student\* temp = head;

if (head == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

while (temp != NULL) {

if (temp->roll == roll\_search) {

cout << "RECORD OF ROLL NO : " << roll\_search << " Found" << endl;

break;

}

else {

temp = temp->next;

}

}

}

}

void search\_by\_node(int node\_search) {

student\* temp = head;

int count\_ = 1;

if (head == NULL) {

cout << "NO NODE IN THE LIST" << endl;

}

else {

if (node\_search <= count) {

while (temp != NULL) {

if (count\_ == node\_search) {

cout << "DISPLAYING ROLL NO : " << temp->roll;

break;

}

else {

temp = temp->next;

count\_++;

}

}

}

else {

cout << "THE VALUE ENTERED IS GREATER THAN THE EXISTING NUMBER OF NODES" << endl;

}

}

}

void insert\_after\_node(student\* newnode, int node\_num) {

int count\_ = 1;

student\* temp = head;

if (head == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

if (node\_num <= count) {

while (temp != NULL) {

if (count\_ == node\_num) {

student\* temp\_next = temp->next;

temp->next = newnode;

newnode->next = temp\_next;

newnode->prev = temp;

count++;

break;

}

else {

temp = temp->next;

}

}

}

}

}

void delete\_at\_head() {

if (head == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

student\* temp\_del = head;

head = head->next;

head->prev = NULL;

delete(temp\_del);

count--;

}

}

void delete\_at\_end() {

student\* temp = head;

student\* prev\_node = head;

if (head == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

while (temp->next != NULL) {

prev\_node = temp;

temp = temp->next;

}

prev\_node->next = NULL;

}

count--;

}

void delete\_before\_node(int del\_node\_opt) {

student\* temp = head;

int count\_ = 1;

if (head == NULL) {

cout << "THE LIST IS EMPTY" << endl;

}

else {

if (del\_node\_opt <= count) {

while (temp != NULL) {

if (count\_ == del\_node\_opt) {

temp->prev->prev->next = temp;

count--;

break;

}

else {

temp = temp->next;

count\_++;

}

}

}

}

}

};

int main() {

list l;

cout << "WELCOME TO STUDENT'S PORTAL \*\*[MADE USING DOUBLY LINKED LIST]\*\*" << endl;

while (1) {

int opt;

cout << "ENTER 1 TO INPUT DATA [AT HEAD]" << endl;

cout << "ENTER 2 TO INPUT DATA [AT END]" << endl;

cout << "ENTER 3 TO DISPLAY DATA FROM HEAD" << endl;

cout << "ENTER 4 TO DISPLAY DATA FROM END" << endl;

cout << "ENTER 5 TO SEARCH BY ROLL NO" << endl;

cout << "ENTER 6 TO SEARCH BY NODE" << endl;

cout << "ENTER 7 TO INPUT DATA AT AFTER NODE" << endl;

cout << "ENTER 8 TO DELETE DATA [AT HEAD]" << endl;

cout << "ENTER 9 TO DELETE DATA [AT END]" << endl;

cout << "ENTER 10 TO DELETE DATA BEFORE NODE" << endl;

cout << "ENTER 11 TO EXIT" << endl;

cin >> opt;

system("CLS");

switch (opt) {

case 1:

{

student\* newnode = l.create\_node();

l.input\_data(newnode);

l.insert\_at\_head(newnode);

system("CLS");

break;

}

case 2:

{

student\* newnode = l.create\_node();

l.input\_data(newnode);

l.insert\_at\_end(newnode);

system("CLS");

break;

}

case 3:

{

l.display\_from\_head();

break;

}

case 4:

{

l.display\_from\_end();

break;

}

case 5:

{

int roll\_search;

cout << "ENTER THE ROLL NO TO BE SEARCHED : "; cin >> roll\_search; cout << endl;

l.search\_by\_roll(roll\_search);

break;

}

case 6:

{

int node\_search;

cout << "ENTER THE NODE NUMBER TO DISPLAY RECORD OF : "; cin >> node\_search; cout << endl;

l.search\_by\_node(node\_search);

break;

}

case 7:

{

student\* newnode = l.create\_node();

l.input\_data(newnode);

int node\_num;

cout << "ENTER THE NODE NUMBER TO INSERT AFTER"; cin >> node\_num; cout << endl;

l.insert\_after\_node(newnode, node\_num);

break;

}

case 8:

{

l.delete\_at\_head();

break;

}

case 9:

{

l.delete\_at\_end();

break;

}

case 10:

{

int del\_node\_opt;

cout << "ENTER NODE THAT YOU WANT TO DELETE"; cin >> del\_node\_opt;

l.delete\_before\_node(del\_node\_opt);

system("CLS");

break;

}

case 11:

{

exit(0);

break;

}

}

}

}

